A Tiling Approach to Network Code Design for Wireless Networks

Michelle Effros Tracey Ho Sukwon Kim
Dept. of Electrical Eng., 136-93, California Institute of Technology, Pasadena, California 91125 USA,
Email: {effros, tho, sukwon}@caltech.edu

Abstract— **We describe a new tiling approach for network code design. The proposed method applies dynamic programming to find the best strategy among a restricted collection of network codes. We demonstrate the proposed strategy as a method for efficiently accommodating multiple unicasts in a wireless coding environment on a triangular lattice and discuss its generalization.**

I. Introduction

We consider the problem of information flow in wireless mesh networks. Each coding network in the family of problems considered contains wireless nodes arranged on the vertices of a triangular lattice (see Figure 1). The network is completely wireless, so each node can communicate only through transmission and receipt of wireless broadcasts. A given wireless node can broadcast information only to nodes in its six neighboring locations. Similarly, a given wireless node directly receives all transmissions sent by its immediate neighbors and no transmissions sent by nodes at a greater distance.

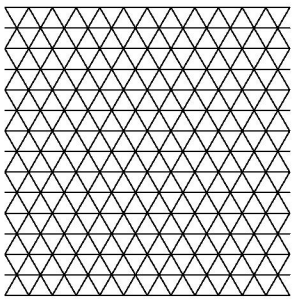

Fig. 1. The nodes of our network lie on the vertices of a triangular lattice.

Even for transmission of independent messages, network coding provides a variety of potential energy saving benefits in wireless networks. First, network coding is useful for information exchange [1], [2], as illustrated in Figure 2(a). In the given example, node v_1 wishes to communicate a single packet of information to node v_3 while node v_3 wishes to communicate a single packet to node v_1. We label these packets as $x_{1,3}$ and $x_{3,1}$, respectively. Without network coding, meeting the given pair of demands requires four transmissions: each source transmits to node v_2 (2 transmissions), and then v_2 transmits first one and then the other message to its intended sink (2 more transmissions). With network coding we obtain a savings in energy – here simply measured by counting the number of transmissions required. The savings is achieved because in this case node v_2 can pass along both messages in a single coded transmission. Precisely, nodes v_1 and v_3 transmit packets $x_{1,3}$ and $x_{3,1}$ to node v_2 (two transmissions); node v_2 takes the bit-wise binary sum of its two received packets ($x_{1,3} \oplus x_{3,1}$) and then broadcasts the sum (one transmission). Since both nodes v_1 and v_3 are within transmission range of node v_2, both receive the mixture packet ($x_{1,3} \oplus x_{3,1}$). Node v_1 decodes $x_{3,1}$ by taking the binary sum of $x_{1,3} \oplus x_{3,1}$ and its known value of $x_{1,3}$. Node v_3 similarly knows $x_{3,1}$ and receives $x_{1,3} \oplus x_{3,1}$, from which it can decode $x_{1,3}$.

The given strategy generalizes from single packet transmissions across a two-hop network to information flows across a path with arbitrarily many hops. We call this strategy *reverse carpooling*. We use the word "carpooling" because the method allows two messages to effectively share a ride through the network: After an initial set-up period, every time an internal node transmits, it transmits a bit-wise binary sum of the next packet that it intends to send forward and the next packet that it intends to send backward along the path. For long paths and long sequences of packets the savings approaches a factor of two. We call it "reverse" carpooling because the strategy only applies when the information flows that want to share a ride are traveling opposite directions.

In addition to the reverse carpooling advantage, network coding is useful at network cross roads, as illustrated in Figure 2(b). Here a single packet ($x_{1,4}$) passes from node v_1 to node v_4, another ($x_{3,6}$) from node v_3 to node v_6, and a third ($x_{5,2}$) from node v_5 to node v_2, The routing solution requires a minimum of six transmissions as each node transmits its known packet to node v_7, which then sends each message along separately. In the network coding solution, here called *star coding*, node v_7 finds the bit-wise binary sum $x_{1,4} \oplus x_{3,6} \oplus x_{5,2}$ and sends that value to all three receivers in a single broadcast transmission. In this case, node v_2 overhears node v_1's transmission of $x_{1,4}$ simply because it is one of v_1's neighbors; it likewise overhears $x_{3,6}$ due to its proximity to v_3. Node v_2 can therefore combine its overheard messages with the coded packet $x_{1,4} \oplus x_{3,6} \oplus x_{5,2}$ to decode the desired message $x_{5,2}$. Nodes v_4 and v_5 likewise overhear the messages that don't interest them and use those to decode their desired packets from $x_{1,4} \oplus x_{5,2} \oplus x_{3,6}$.

Each application of the star coding strategy gives a savings of two transmissions. The same approach likewise applies when only two paths cross provided that the cross config-

1-4244-0036-8/06/$20.00 ©2006 IEEE

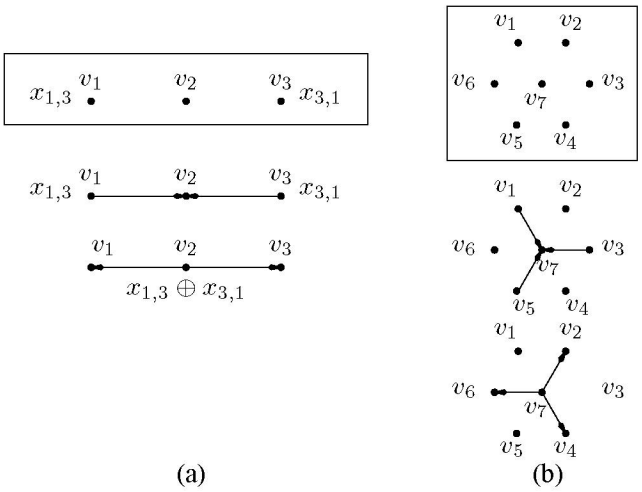

Fig. 2. Two examples where network coding provides an energy savings relative to routing in a wireless network.

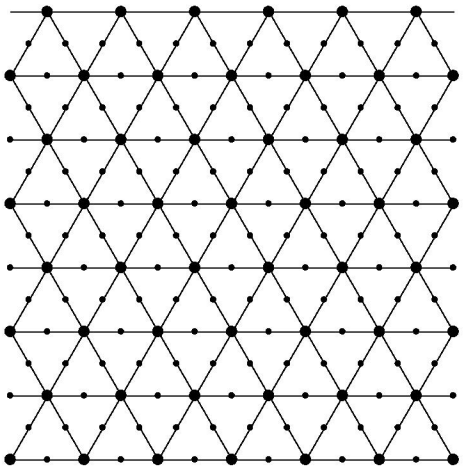

Fig. 3. The network from Example 1. Each path from transmitter to receiver takes a single straight line from one edge of the network to another. For every line, there are two information fbws – one in each direction.

uration allows each of the intersection outputs to overhear the message that it doesn't require. The savings from such a crossing, here called a 2-star, is a single transmission.

We here consider network code design for multiple unicast sessions on the given triangular lattice. In order to simplify code design, we impose the constraint that the network code to be designed is a combination of only (2- or 3-input) star coding nodes, carpooling nodes, and routing nodes. We call the resulting family of possible network codes SCoR codes. While SCoR codes are a restrictive subset of the family of network codes, the family is sufficiently rich both to allow significant performance gains and to raise interesting questions about code design. For example, the following SCoR code gives a multiplicative advantage approaching 2.4 over the optimal routing solution – that is, routing alone requires 2.4 times as many transmissions as the described SCoR code. Following this example, we suggest a variety of interesting code design questions related to code design in this restricted network coding environment.

Example 1: Consider the network of Figure 3. Fifty-two distinct flows traverse the network, with one passing in each direction along each straight line from one end of the network to the other. Every node of the network is active and, after an initial set-up period, serves either two or three receivers with each transmission. Every reverse carpooling point serves two receivers with each transmission, at each use sending the bit-wise sum of a single packet passing from left to right and a single packet passing from right left through the network. Every star node alternates between two codes, thereby operating twice as frequently as any neighboring carpool node. Precisely, each star node alternates between a star code that takes the form of Figure 2(b) and a star code that reverses the roles of the input- and output-nodes in that example. Note that in the given SCoR code, any node v can independently decode all messages intended to traverse any path through v. However, in the proposed code, star nodes code their received signals directly while carpool nodes decode before re-encoding and transmitting their outgoing packets.

To calculate the benefit of the given code, we notice that passing one packet of information in each direction along each path requires six transmissions at each intersection point and two transmissions at each non-intersection point. With coding, each intersection point requires two transmissions while each non-intersection point requires one transmission. Since (excluding the edges of the network) one quarter of the nodes are intersection points and three quarters are non-intersection points, the ratio of the number of transmissions required with routing to the number of transmissions required without routing is

$$\frac{(1/4) * 6 + (3/4) * 2}{(1/4) * 2 + (3/4) * 1} = 2.4.$$

Thus extending the network indefinitely (to send the fraction of perimeter nodes to zero) and sending longer and longer sequences of packets (to send the cost of initialization to zero) gives an asymptotic factor of 2.4 improvement over the pure routing solution.

The above argument suggests an upper bound of 3 on the multiplicative advantage of SCoR codes. Achieving this advantage precisely would require star coding at all nodes of the network. Unfortunately, the decoding algorithm seems to break down under these conditions. □.

The previous example both motivates the use of SCoR codes and raises a variety of questions. The given example sends every message along the shortest path from its source node to its terminal node. Is it possible to achieve an energy savings in wireless network coding using non-shortest-path solutions? The given example sends every message down only a single path from its source node to its terminal node. Is it possible to achieve an energy savings in wireless network coding using multipath solutions? Surprisingly, both of these techniques can give advantages in the given coding environment, as we

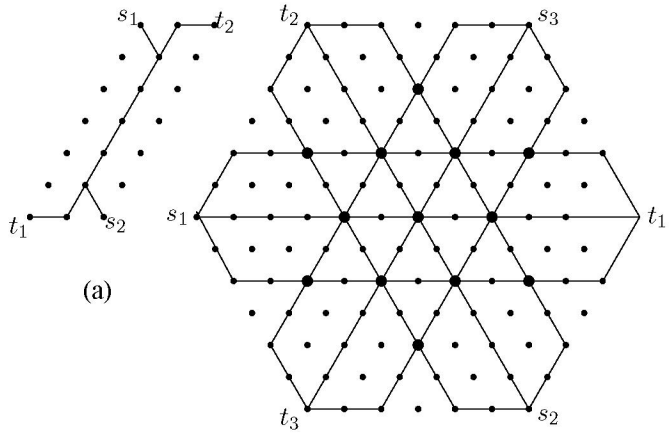

Fig. 4. The networks for (a) Example 2 and (b) Example 3. (a) Deviations from the shortest path sometimes create coding opportunities and performance improvements. (b) Multipath coding may be useful even when each source is connected to its corresponding terminal by a unique shortest path.

illustrate with the following examples.

In the discussion that follows, we describe a network's $k \geq 1$ demands with the vectors $(s_1, t_1, w_1), \ldots, (s_k, t_k, w_k)$, where demand i is specified by the node s_i where the source originates, and the node t_i where the source is required. It will be useful in later discussions to specially identify any pair of demands that is identical except for a reversal of the roles of transmitter and receiver (e.g., $s_1 = v$, $t_1 = v'$ and $s_2 = v'$ and $t_2 = v$) and treat any such pair of demands as a single two-way communication rather than a pair of one-way communications. We therefore pair up any opposite demands and use the index $w_i \in \{1, 2\}$ to specify one- or two-way communication.

Example 2

Let $(s_1, t_1, 1)$ and $(s_2, t_2, 1)$ be the demands for the simple network of Figure 4(a). Each source is connected to its terminal by a 6-edge shortest path. The shortest paths do not intersect, and thus the total cost of communication without coding is 12 transmissions. Taking each message off of its shortest path creates an opportunity for reverse carpooling. The combined cost of the two detours is two transmissions, and the carpooling savings is five transmissions per bit, giving a net asymptotic savings of three transmissions per bit end-to-end for deviating from the shortest path. The benefit increases as the length of the carpooling opportunity grows.

Example 3

The hexagonal network of Figure 4 demonstrates that sometimes multi-path coding gives an energy savings over single-path coding. While the gains are extremely small in this example, they open the door for greater benefits through larger examples. In this case, the benefit arises since more paths create more opportunities for star coding.

II. Algorithm

We begin by describing an optimal algorithm for SCoR code design on a small hexagonal network. Since the hexagon tiles the triangular lattice, we then apply a dynamic programming argument to build SCoR codes for larger and larger networks by tiling together solutions for smaller networks. While the complexity of the algorithm makes it impractical for very large networks, solutions for the tractable cases may lend insights useful in networks of all sizes.

A. Optimal SCoR Coding in a Hexagonal Network

Consider the small, hexagonal network shown in Figure 5(a). For any integer $k \geq 1$, define demands

$$(s_1, t_1, w_1), \ldots, (s_k, t_k, w_k),$$

where s_i and t_i are, by assumption, distinct nodes on the edges of the given hexagon ($s_i, t_i \in \{v_1, \ldots, v_6\}$). We wish to find the best SCoR code that satisfies demands $(s_1, t_1, w_1), \ldots, (s_k, t_k, w_k)$.

Since the network is very small, its solution is very simple, no matter what the collection of demands. The following principles are useful for solving any instance of this problem.

1) *Shortest path solutions are sufficient for optimality.* Given the size of the network, the shortest path from s_i to t_i passes through at most one intermediate node. Any detour off of that shortest path costs at least one transmission and gains at most one transmission in savings. Thus for any non-shortest-path solution there exists a shortest path solution that does equally well.
2) *All coding may be performed at the network's central node.* Given the previous observation, we need only show that messages passing each other on the edge of the network can be coded equally well at the center of the network. For a coding opportunity to arise, each shortest path under consideration must have length 2 (length-one paths leave no opportunity for coding, while length-three paths along the edge of the network cannot be shortest paths). Thus all coding opportunities at edge vertices are functionally equivalent to the reverse carpool operation that can be applied when one message passing from v_1 to v_3 via v_2 meets another message passing from v_3 to v_1 via v_2. For any such network code, there exists an alternative network code with the same performance that codes at the central node rather than the perimeter node v_2. Precisely, passing each message from its source to its destination via node v_7 rather than node v_2 allows us to move the coding operation to the central node. This observation leads immediately to the following principle for code design.
3) *When the shortest path is not unique, the alternative that traverses the central node is always sufficient for optimality.*

Given these observations, the problem of optimal code design on the hexagonal network is easily solved for any collection of demands. The algorithms first routes each demand along its shortest path, choosing the path through the center when the shortest path is not unique. The algorithm then counts coding opportunities by removing first 3-stars,

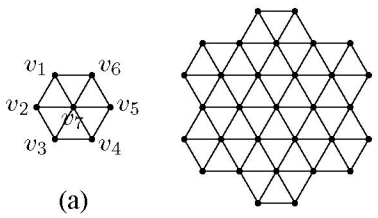

Fig. 5. (a) A seven-node hexagonal network. (b) A larger network created by tiling together 7 hexagonal networks.

then 2-stars and carpools. The resulting solution is an optimal SCoR code for any k and any collection of demands $(s_1, t_1, w_1), \ldots, (s_k, t_k, w_k)$.

B. Code Design for Larger Networks

In the discussion that follows, we recursively use code designs on small networks to build codes for larger and larger networks. The smallest network is the hexagonal network of Figure 5(a). The next larger network is the seven-hexagon network shown in Figure 5(b). The following network likewise comprises of seven copies of the seven-hexagon network, and so on. To describe these networks recursively, let $\mathcal{N}_1$ be the hexagonal lattice of Figure 5(a). For each h greater than one, network $\mathcal{N}_h$ comprises one central and 6 surrounding copies of the network $\mathcal{N}_{h-1}$. While only $\mathcal{N}_1$ is a true hexagon, we refer to all $\mathcal{N}_h$ as hexagons since each takes roughly that shape and, like hexagons, each tiles the triangular lattice.

The previous section describes, for any k, the optimal solution to all k-demand network coding problems on network $\mathcal{N}_1$. We next apply this family of solutions within a dynamic programming argument to build codes for successively larger $\mathcal{N}_h$.

A few restrictions are required in order to practically build SCoR codes for larger networks by simply tiling together solutions for smaller networks. First, we prohibit most coding operations at nodes lying along the perimeter of the given hexagon. In particular, we allow only reverse carpool coding at the perimeter nodes and allow that only when one two-way demand connects to another two-way demand at the given node. We allow coding in this case because we can recognize the opportunity for coding here from the demands alone. We prohibit other forms of coding at perimeter nodes because in other cases determining whether coding is possible would require knowledge not just of the network's demands but also details of how those demands are met within the network. Second, we restrict our code design by enforcing the constraint that no single flow enters or leaves a single hexagon more than once. This constraint does not prohibit the investigation of multipath solutions; any multipath solution can be represented by treating distinct paths as separate demands that share the same transmitter and receiver. It does, however, restrict the number of demands required on each subnetwork to be no greater than the total number of demands in the network. Finally, we constrain each flow on network $\mathcal{N}_h$ to pass through at most seven of the seven subnetworks $\mathcal{N}_{h-1}$ that comprise it.

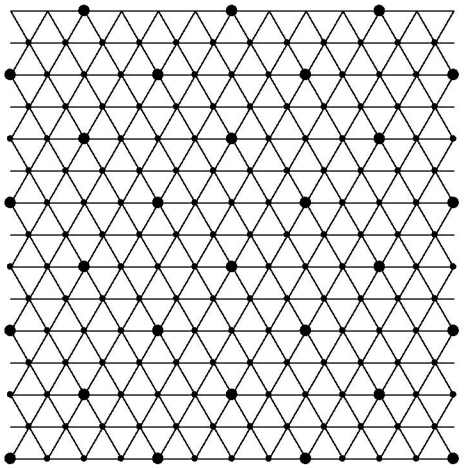

Fig. 6. A modification of Example 1 that is consistent with restricted SCoR coding. As in the previous example, each path from a transmitter to its receiver takes a single straight line from one edge of the network to another. Again, all demands are two-way links. The density of demands is higher here than in Example 1.

We call a SCoR code that conforms to these restrictions a *restricted SCoR code*. While the restricted SCoR code for any collection of demands on hexagon $\mathcal{N}_1$ is an optimal SCoR code, that is not necessarily true in general. Figure 6 gives a variation on Example 3 that is consistent with restricted SCoR coding. Only the nodes labeled by larger circles (the hexagon centers) perform star coding; each alternates between the two distinct star codes. All other nodes alternate between three distinct reverse carpooling codes – one for each angle. The asymptotic multiplicative coding advantage of this restricted SCoR code relative to the optimal routing solution is $6/((1/4)*2+(3/4)*3) = 24/11 \approx 2.18$.

The algorithm $rSCoR(h,k)$ finds the optimal restricted SCoR code for *every* possible set of at most k demands (s_i, t_i, w_i), where for each demand, s_i and t_i are distinct nodes on the outer perimeter of $\mathcal{N}_h$. The algorithm's output is a table describing the number of transmissions required for every possible collection of at most k demands on $\mathcal{N}_h$. Running $rSCoR(1,k)$ applies the algorithm of Section II-A for each possible set of demands on network $\mathcal{N}_1$. For any $h \geq 1$, $rSCoR(h,k)$ proceeds by first running $rSCoR(h-1,k)$. The algorithm then tiles together matching solutions on $\mathcal{N}_{h-1}$ to form solutions on $\mathcal{N}_h$. A set of 7 solutions to $\mathcal{N}_{h-1}$ matches if and only if (a) its flows are conserved (every flow leaving one node must enter one and only one of its neighbors) and (b) it meets some collection of at most k demands on the perimeter of the network $\mathcal{N}_h$. A legitimate tiling is any tiling with matching solutions in which each flow on network $\mathcal{N}_h$ passes through at most four of the seven copies of $\mathcal{N}_{h-1}$ that comprise it. The cost of a legitimate tiling is the sum of the cost of its tiles minus the number of times each two-way flow crosses from one copy of $\mathcal{N}_{h-1}$ to another (since the

shared node for this crossing has a new opportunity for reverse carpool coding). The algorithm records the lowest cost tiling for each collection of at most k demands on the perimeter of the network.

The complexity of $rSCoR(h,k)$ may be loosely bounded as follows. Each of the at most k demands enters and leaves at one of the $6(3^h)$. By assumption, each demand passes through at most four of the seven subnetworks of $\mathcal{N}_h$, entering and leaving each subnetwork at most once. Thus associated with each path are at most three transitions between subnetworks, each of which may occur at any one of the 3^h+1 nodes shared by those subnetworks. The resulting complexity is $O(3^{5hk})$. A single application of this program solves all problems of at most k demands on a network of size $O(7^h)$.

III. Conclusions and Future Work

The proposed algorithm uses a dynamic programming argument to build network codes for larger networks by tiling together network codes from smaller networks. While the complexity of the algorithm makes its solution for very large numbers of h and k, solutions for tractable values of k may give insights into general strategies for network code design on wireless networks.

Future experimental work is expected to help answer basic questions about the network coding in wireless networks. What is the relative merit of shortest- and non-shortest path coding? How do the performance of single- and multi-path coding compare?

In addition, a number of algorithmic improvements are possible. The given network construction leads to networks with increasingly irregular boundaries. The restriction that any path enter and leave a subnetwork $\mathcal{N}_{h-1}$ may be damaging under these conditions. An alternative version of the algorithm, based on tiling the space with parallelograms rather than hexagons, may yield a scenario where optimal solutions are guaranteed to enter and exiting subnetworks at most once. The result may be better solutions at similar costs. Likewise, it may be interesting to investigate bounds on the number of subnetworks visited by an arbitrary optimal flow. Investigations of this type are expected to lead to improvements in both algorithmic efficiency and performance.

Acknowledgment

This work was supported in part by the National Science Foundation under grant number CCF-0325324 and by Caltech's Lee Center for Advanced Networking.

References

[1] Y. Wu, P.A. Chou, and S.-Y. Kung. Information exchange in wireless networks with network coding and phyical-layer broadcast. Technical Report MSR-TR-2004-78, Microsoft Research, Redmond, WA, August 2004.

[2] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard. The importance of being opportunistic: practical network coding for wireless enviornments. In *43rd Allerton Annual Conference on Communications, Control, and Computing*, Monticello, IL, September 2005. IEEE. Invited paper.